Scala Fire Tower

Università di Bologna · Campus di Cesena

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Alessandro Becci	0001034968
Luca Tonelli	0001034968
Riccardo Battistini	0001034968

1. Introduzione (INSIEME)

2. Introduzione

Scala Fire Tower è un gioco da tavolo competitivo in cui i giocatori gestiscono torri di avvistamento antincendio, cercando di proteggere la loro torre dagli incendi forestali che si diffondono, mentre cercano di dirigere le fiamme verso le torri degli avversari. I giocatori utilizzano risorse come vigili del fuoco, acqua e linee tagliafuoco per controllare la diffusione del fuoco. La direzione del fuoco è influenzata dal vento e i giocatori possono cambiare la direzione del vento usando le carte azione. L'obiettivo è eliminare gli avversari bruciando le loro torri. Il gioco può essere giocato da 1 a 2 giocatori e di solito dura 15 minuti.

3. Requisiti (INSIEME)

4. Requisiti

https://docs.asciidoctor.org/asciidoc/latest/syntax-quick-reference/ https://docs.asciidoctor.org/asciidoc/latest/

4.1. Requisiti di business

Regolamento ufficiale

- Il gioco deve permettere lo svolgimento di una partita, interagendo con le entità del gioco e in accordo con il regolamento ufficiale del gioco.
  - Possibilità di piazzare dei token fuoco o tagliafuoco sulla griglia.
  - Possibilità di pescare e scartare carte e di scartare una carta dopo che è stata giocata.
  - Possibilità di applicare pattern di diverso tipo a seconda del tipo di carta giocata.
  - Possibilità di modificare la direzione del vento casualmente o in base alla carta giocata.
  - Possibilità di utilizzare carte normali e carte speciali per interagire col gioco

4.2. Requisiti funzionali

- Utente
  - L'utente deve visualizzare un menu iniziale dalla quale è possibile iniziare una partita settandone i parametri.
  - L'utente deve poter visualizzare la schermata del gioco comprensiva di tutte le sue parti.
  - L'utente deve poter interagire con la griglia per piazzare i pattern.
  - L'utente deve poter cliccare su una carta ottenendo i pattern possibili sulla griglia.

    - L’utente deve poter compiere una scelta nel caso di carte multi-scelta.
    - L’utente deve poter interagire con la griglia per piazzare i pattern.
    - L’utente deve poter scartare le carte.
    - L’utente deve poter scegliere di finire il turno senza giocare una carta speciale.
    - L’utente deve poter vincere quando piazza il fuoco sulla posizione di una torre nemica.
- Sistema
    - Il sistema deve gestire le schermate di menu e di gioco.
    - Il sistema deve visualizzare la scena del gioco, composta di griglia, mano, direzione del vento, informazioni sulla partita, dado e scarta carte.
    - Il sistema deve calcolare i pattern disponibili sulla griglia per ogni tipo di effetto di carta disponibile.
    - Il sistema deve offrire un sistema di aggiornamento legato alle azioni compiute dal giocatore.
    - Il sistema deve gestire il comportamento del nemico (bot).

4.3. Requisiti non funzionali

- Grafica: il gioco deve risultare simile a quello originale visivamente.
- Usabilità: il gioco deve far sì che l’utilizzo delle carte sia il più facile possibile.

4.4. Requisiti di implementazione

- Utilizziamo:
    - Scala 3.4.2
    - ScalaTest X.X.X
    - JDK 17+

5. Design architetturale (INSIEME)

Descrizione del pattern architetturale utilizzato (MVC + Cake)

6. Design di dettaglio

7. Design di dettaglio

In questa sezione, si mostrano le scelte di design più critiche circa l’organizzazione della struttura del sistema e dei suoi componenti.

7.1. Model

7.1.1. Gameboard

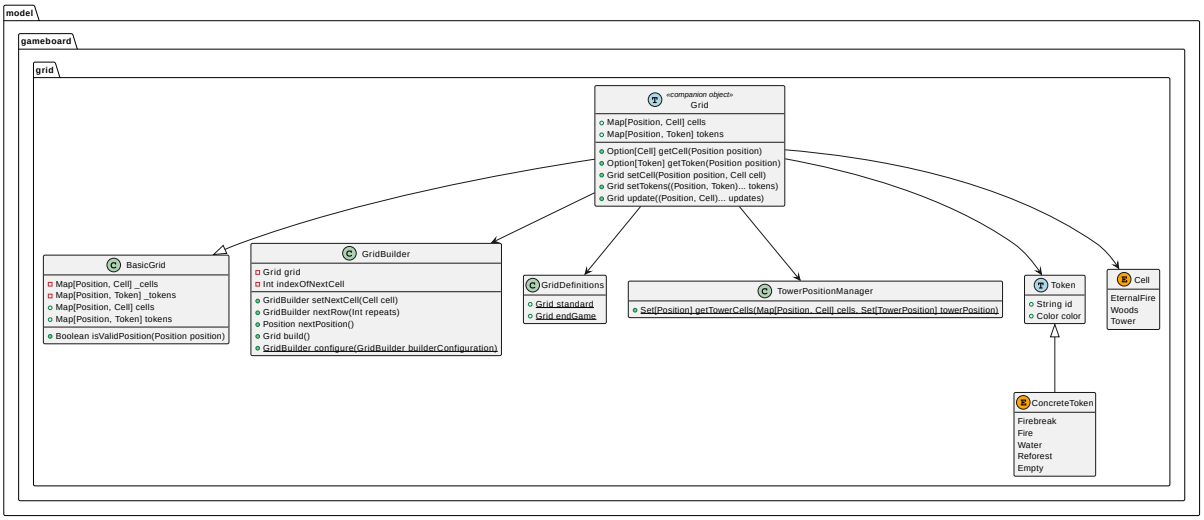

diagramma degli stati

diagramma di sequenza

diagramma delle classi

7.2. Model

divisione del turno in fasi - TONE effetti/carte - RICK gameboard, grid, DSL - ALE player (bot) - ALE, TONE

prolog attacco e difesa - ALE carte (availablePattern) - RICK scelta pattern - TONE

7.3. View

inizializzazione della view - RICK componente GameComponent - RICK griglia (hover, click) - ALE mano - TONE deck - TONE sidebar RICK vista menu - TONE

7.4. Controller

observable e subscriber Bot - ALE Internal - TONE Model - ALE View - RICK

protocolli di scambio dei messaggi

Activation ALE Discard / PlayCardController / TONE Refresh RICK

7.5. Organizzazione del codice/Struttura dei package

diagramma dei package

8. Retrospettiva

8.1. Processo di sviluppo

8.1.1. Meeting e interazioni pianificate

8.1.2. Modalità di divisione dei task

8.1.3. Modalità di revisione dei task

8.1.4. Strumenti di test

8.1.5. Workflow Git

8.1.6. Strumenti di qualità del codice

8.1.7. Strumenti di CI/CD

8.2. Suddivisione del lavoro

8.2.1. Alessandro Becci

8.2.2. Luca Tonelli

8.2.3. Riccardo Battistini

9. Sviluppi futuri (INSIEME)

10. Guida Utente (INSIEME)